文字溢出显示

1.如果文字显示不开也必须强制一行内显示

white-space: nowrap;

2.溢出部分隐藏起来

overflow: hidden;

3.文字溢出的时候用省略号显示

text-overflow: ellipsis;

```
<style>
  div {
    width: 200px;
    height: 20px;
    background-color: pink;
    margin: 100px auto;
    /* 文字显示不开,自动换行 */
    /* white-space: normal; */

    /* 包括3个部分:  */

    /* 1.如果文字显示不开也必须强制一行内显示 */
    white-space: nowrap;

    /* 2.溢出部分隐藏起来 */
    overflow: hidden;

    /* 3.文字溢出的时候用省略号显示 */
    text-overflow: ellipsis;
       }
   </style>
</head>

<body>
    <div>可是刺激市场价看上，是哦看可是你老是看你开始</div>
</body>

```

效果展示:

可是刺激市场价看上，是...

响应式布局

媒体查询-响应式布局

@media 声明媒体查询

screen 屏幕类型

关键词 and

条件 min-width max-width

引入方式：

1 直接在css环境中使用

2 使用link标签引入@media属性

```
引入媒体查询的两种方式：
 1.直接在css环境里面
   @media screen and (max-width:900px) {
     css代码
 }
2.在link标签里面增加  media属性
<link rel="stylesheet" href="./css/css2.css" media="screen and (min-width:901px)">
```

sass

css编译器，开发速度更加快，可读性和可维护性更高

1、Scss格式

Scss格式，使用大括号“{}”和分号“;”，也就是类似CSS书写的格式。

举例：

```
$myColor:white
$bgColor:red
body
{
    color:$myColor
    background-color:$bgColor
}
```

2、变量的应用

声明变量：使用$

变量命名的规范：

1.可以使用字母、数字、下划线、中横线

2.数字不能开头 $1myfont 错误写法(数字不能开头)

3.不区分下划线和中横线 $you-color == $you_color

4.驼峰命名法(第一个单词的首字母小写，后面单词的首字母大写)

在Sass中，变量类型：

（1）数字值，如10、10px、-10、1.1等；

（2）字符串，如 "我是字符串"、sans-serif等；(用引号引起来的被称为字符串)

（3）布尔值，true、false；

（4）颜色值，如

16进制 #FF00FF

rgb (255,0,255)

hsl (360,50%,50%)；

(5) 列表

(6) null 值

变量作用域问题

1.全局变量 直接定义在scss文件里面

2.局部变量 定义在局部作用域里面(用{}扩起来的被称为局部作用域)

Sass 变量的作用域只能在当前的层级上有效果。

```scss
//全局变量
$myColor: red;

h1 {
  $myColor: green;   /* 只在 h1 里头有用，局部作用域 */
  color: $myColor;
}

p {
  color: $myColor;
}
```

将以上代码转换为 CSS 代码，如下所示：

```
h1 {
  color: green;
}

p {
  color: red;
}
```

3.!global 让局部变量变为全局变量

注意:局部变量提升为全局变量必须写在 使用的前面

否则依然会提示未定义该变量(代码从上往下执行)

```
// 全局变量
$myHeight:200px;

.demo {
    // 局部变量
    $myWidth:300px !global;
    //!global  让局部变量变为全局变量

    height: 200px;
    width: 200px;
    // background-color: $myColor;
    // background-color: $myBackColor; 错误用法  变量使用在全局提升之前
}

.demo2 {
    // 局部变量
    $myBackColor:green !global;

    height: $myHeight;
    width: $myWidth;
    // width: $myWidth; 错误用法，demo2访问不了demo里面的局部变量
    background-color: red;
}

p {
    color: $you_color;
```

```
}
```

3、sass的引用

引用css文件而不是scss文件

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <!-- 引入css文件不是scss文件 -->
    <link rel="stylesheet" href="./sass/index.css">
</head>

<body>
    <div class="demo"></div>
    <p>我是sass</p>
    <div class="demo2"></div>
</body>
```

4、sass的嵌套规则

（1） 与html结构保持一致，会被识别成后代选择器， 选择器嵌套

（2）属性嵌套；

（3）伪类嵌套；

要想使用子代 在前面加 >

要想使用:hover 在前面加 &

&表示当前的选择器

sass代码：

```
.nav {
    height: 40px;
    background-color: $navColor;

    .navBox {
        display: flex;
        align-items: center;
```

```
8
9           .nb-left {
10              line-height: 40px;
11
12              a {
13                  color: #b0b0b0;
14                  font-size: 12px;
15                  &:hover {
16                      color: #fff;
17                  }
18              }
19
20              span {
21                  font-size: 12px;
22                  color: #424242;
23              }
24          }
25      }
26  }
```

css转译代码：

```
1  .nav {
2    height: 40px;
3    background-color: #333333;
4  }
5
6  .nav .navBox {
7    display: flex;
8    align-items: center;
9  }
10
11 .nav .navBox .nb-left {
12   line-height: 40px;
13 }
14
15 .nav .navBox .nb-left a {
16   color: #b0b0b0;
17   font-size: 12px;
```

```
}

.nav .navBox .nb-left a:hover {
  color: #fff;
}

.nav .navBox .nb-left span {
  font-size: 12px;
  color: #424242;
}

```

5、sass文件的引入

在scss文件引入sass文件时，直接引入scss文件（没有后缀）

格式：

@import "scss文件（无后缀名)"

sass的 '@import' 规则在生成css文件时就把相关文件导入进来

Partials，它们不会被编译成css文件

引入sass文件代码：

```
@import "1";
@import "2";


.demo {
    height: $myHeight;
    width: $myWidth;
    background-color: $myColor;
}
```

css转译代码：

```
* {
  margin: 0;
  padding: 0;
}

```

```
a {
  text-decoration: none;
}

li {
  list-style: none;
}

.demo {
  height: 400px;
  width: 300px;
  background-color: green;
}

```

6、sass混入

格式:

@mixin 声明混入

格式 @mixin name {

}

@include 使用混入

格式: @include name()

```
@mixin sanJiao {
    height: 0;
    width: 0;
    border: 50px solid transparent;
    border-bottom-color: green;
}

.demo {
   @include sanJiao()
}
```

混入传值:

@mixin name(变量名,变量名:默认值)

@include name(传入的值,传入的值)

```scss
@mixin sanJ($width,$color:red) {
    height: 0;
    width: 0;
    border: $width solid transparent;
    border-bottom-color: $color;
}

.demo {
    @include sanJ(200px,pink)
}
```

继承、占位符和混入的声明方式和调用方式

方法	声明方式	调用方式
继承	.class	@extend
占位符	%placeholder	@extend
混入	@mixin	@include

7、sass的继承

Sass中，我们可以使用“@extend”来继承一个样式块，从而实现代码的重用

使用sass的时候，最后一个**减少重复**的主要特性就是选择器继承。选择器继承是说一个选择器可以继承另一个选择器定义的所有样式。这个通过 **@extend** 语法实现

```scss
/* 通过选择器继承继承样式 */
.father {
  border: 1px solid red;
  background-color: #fdd;
}
.son {
  @extend .father;
  border-width: 3px;
}
```

.son不仅会继承.father自身的所有样式，任何跟.father有关的组合选择器样式也会被.son以组合选择器的形式继承。

```css
/*.son从.father继承样式*/
.father a{  /*应用到.son a*/
  color: red;
  font-weight: 100;
}
h1.father { /*应用到h1.son*/
  font-size: 1.2rem;
}
```

布局

1、rem

1.单位

px 像素

em 相对于父元素的字体大小

rem 相对于根标签的字体大小(根标签就是html标签)

根标签里面默认字体大小为16px

rem值 = 页面元素值（px） / html里 font-size 字体大小

rem的优点就是可以通过修改html里面的文字大小来改变页面中元素的大小可以整体控制

2、meta视口标签

name="viewport" 视口标签

width=device-width 宽度=设备的宽度

initial-scale=1.0 初始缩放比例 1.0表示不缩放 大于0的数字

maximum-scale 允许最大缩放比例 大于0的数字

minimum-scale 允许最小缩放比例 大于0的数字

user-scalable 用户是否可以缩放（yes或no（1或0）

```html
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0 ,user-scalable=0">
```

3 设备像素比

375px / 750px = 1 / 2

4 二倍图

开发环境的像素是实际像素的1/2

一张20*20像素的图片不会被放大到40*40(因为图片本身的像素我们改不了)

为了解决这个问题: 可以把一张40*40像素的图片放在20*20的盒子里面

也就是把40*40像素的图片手动缩小为 20 * 20。将图放到手机里面，手机自动放大2倍变成 40 * 40，这样就不会造成图片模糊

vw 100vw占满整个设备的宽度（例如设备宽375px，设置100vw后，宽度为375px）

vh 100vh占满整个设备的高度 （例如设备高667px，设置100vh后，高度为667px）

5 移动端书写步骤

（1）设置meta视口标签

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0 ,user-scalable=0">

即：

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0 ,user-scalable=0">
```

（2）引入js代码实现rem自适应设备

（3）设置body标签里的font-size的值为：16px

（4）把px转成rem（alt+z)

（5）其他与css书写方式一致

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <!-- 第一步 -->
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">
    <title>移动端使用步骤</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        /* 第三步 */
        body {
            font-size: 16px;
```

```
        }

        /* 第四步 */
        .demo {
            width: 4rem;
            height: 6rem;
            background-color: pink;
        }
    </style>
</head>

<body>
    <div class="demo"></div>

    <!-- 第二步 -->
    <script>
        (function (doc, win) {
            var docEl = doc.documentElement,
                resizeEvt = 'orientationchange' in window ? 'orientationchange' : 'resize',
                recalc = function () {
                    // 宽度
                    var clientWidth = docEl.clientWidth;
                    if (!clientWidth) return;
                    //
                    if (clientWidth >= 750) {
                        docEl.style.fontSize = '100px';
                    } else {
                        // 动态设置html的font-size
                        docEl.style.fontSize = 100 * (clientWidth / 750) + 'px';
                    }
                };
            if (!doc.addEventListener) return;
            win.addEventListener(resizeEvt, recalc, false);
            doc.addEventListener('DOMContentLoaded', recalc, false);
        })(document, window);
    </script>
</body>

</html>
```